Konten Digital Interaktif Basis Data SQL, DDL, dan DML



Disusun Oleh:

1. **Michael Gerrald Liem**
2. **Lie William**
3. **Ferianus**

SMA Talenta

2024

Kata Pengantar

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga ebook berjudul "Konten Digital Interaktif Basis Data *SQL*, *DDL*, dan *DML*" ini dapat terselesaikan dengan baik. *Ebook* ini disusun sebagai upaya untuk memberikan pemahaman yang komprehensif mengenai basis data, khususnya *SQL (Structured Query Language)*, yang meliputi *DDL (Data Definition Language)* dan *DML (Data Manipulation Language)*.

Di era digital saat ini, penguasaan teknologi informasi menjadi suatu keharusan bagi setiap individu yang ingin bersaing di dunia kerja maupun akademis. Basis data merupakan salah satu komponen penting dalam teknologi informasi yang berfungsi untuk menyimpan, mengelola, dan memanipulasi data. Oleh karena itu, pemahaman yang baik mengenai SQL, DDL, dan DML menjadi sangat krusial.

Ebook ini dirancang secara interaktif dengan tujuan memudahkan pembaca dalam mempelajari konsep-konsep dasar hingga lanjutan dalam basis data. Melalui konten digital interaktif, kami berharap pembaca tidak hanya memahami teori, tetapi juga mampu mempraktikkan dan mengimplementasikan pengetahuan yang diperoleh.

Adapun beberapa topik yang dibahas dalam ebook ini meliputi:

- *Structured Query Language* (*SQL*)
- *Data Definition Language* (*DDL*)
- *Data Manipulation Language* (*DML*)

Kami menyadari bahwa dalam penyusunan ebook ini masih terdapat kekurangan dan keterbatasan. Oleh karena itu, kami sangat mengharapkan saran dan kritik yang membangun dari pembaca sekalian demi penyempurnaan edisi berikutnya.

Akhir kata, semoga ebook ini dapat memberikan manfaat yang sebesar-besarnya bagi pembaca dan menjadi referensi yang berguna dalam mengembangkan kemampuan di bidang basis data. Terima kasih kepada semua pihak yang telah memberikan kontribusi dalam penyusunan ebook ini.

Selamat membaca dan semoga sukses!

Bandung, Mei 202

DAFTAR ISI

BAB 1
Perintah Umum *Structured Query Language* (*SQL*)

Structured Query Language (*SQL*) adalah bahasa pemrograman untuk menyimpan dan memproses informasi dalam basis data relasional. Sebuah basis data relasional menyimpan informasi dalam bentuk tabel, dengan baris dan kolom yang mewakili atribut data yang berbeda serta berbagai hubungan antara nilai data. Kita dapat menggunakan pernyataan *SQL* untuk menyimpan, memperbarui, menghapus, mencari, dan mengambil informasi dari basis data.

Berikut adalah Perintah Umum *SQL*:

1. **Show Tables**
   'Show table' dalam bentuk *SQL* adalah perintah yang digunakan untuk menampilkan daftar tabel yang ada dalam sebuah *database*.

KODE:

sql Salin kode

```sql
SHOW TABLES;
```

CONTOH	
Hasil:	`SHOW TABLES` Tables_in_catatan_penjualan_toko_sparepart_motor produk transaksi

2. **Describe**
   'Describe' biasanya digunakan untuk mendapatkan informasi tentang struktur suatu tabel dalam basis data. Perintah ini memberikan deskripsi singkat tentang kolom-kolom dalam tabel, seperti nama kolom, tipe data, ukuran, dan apakah kolom tersebut mengizinkan nilai-nilai yang *null* atau tidak.

KODE:

sql Salin kode

```sql
DESCRIBE nama_tabel;
```

CONTOH
`DESCRIBE produk` **Hasil:**

	Field	Type		Null	Key	Default		Extra
☐	Kode_Produk	int(11)	7B	YES		(NULL)	0K	
☐	Nama_Sparepart	text	4B	YES		(NULL)	0K	
☐	Stok	int(11)	7B	YES		(NULL)	0K	
☐	Tipe	text	4B	YES		(NULL)	0K	
☐	Harga	int(11)	7B	YES		(NULL)	0K	

3. **Use**
   'Use' adalah sebuah perintah dalam *SQL* yang digunakan untuk memilih database tertentu. Biasanya perintah ini digunakan ketika terdapat banyak *database* di dalam *SQL*, dan kita harus menentukan *database* mana yang akan dioperasikan.

KODE:

sql   Salin kode

```sql
USE nama_basis_data;
```

CONTOH
`USE catatan_penjualan_toko_sparepart_motor`

BAB 2
Syarat dan Pengelompokan Data

Where, And, Or, Order, Group, sebuah klausa yang digunakan dalam mengambil, memperbarui, atau menghapus data dari sebuah database dengan cara yang terstruktur dan terkontrol. Klausa ini membantu untuk merancang *query* yang tepat sesuai dengan keinginan.

1. **WHERE**
   'Where' adalah klausa yang digunakan untuk menentukan kriteria yang harus dipenuhi oleh nilai bidang dalam suatu tabel untuk data yang ingin disertakan dalam hasil *query*. Klausa ***'Where'*** memungkinkan untuk memfilter data yang ingin dilihat berdasarkan kriteria yang ditentukan.

KODE:

sql Salin kode

```sql
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

CONTOH

```sql
SELECT * FROM produk WHERE Harga > 500000;
```



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
3	Speedo ...	12B	20	SIP	3B	1500000
6	Knalpot...	15B	2	Samlong	7B	1000000
7	Velg Marus	10B	5	Bintang	7B	30000000
8	ECU	3B	5	RC M...	9B	8000000
9	Head Lamp	9B	3	XO R...	10B	2000000
10	Shockbr...	12B	6	Ohlins	6B	5000000
11	Shockbr...	12B	5	KTC ...	12B	1700000
13	STOP Lamp	9B	8	Luigy	5B	1500000
14	Kaliper	7B	8	RCB	3B	800000

2. **AND**
'And' adalah sebuah operator yang digunakan untuk menggabungkan dua atau lebih kondisi dalam klausa *'Where'*. Operator *'And'* memastikan bahwa semua kondisi yang digabungkan harus dipenuhi oleh nilai bidang dalam suatu tabel untuk data yang ingin disertakan dalam hasil *query*.

KODE:

```sql
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2;
```

CONTOH

```sql
SELECT * FROM produk WHERE tipe = 'Luigy' AND Harga > '500000';
```



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
13	STOP Lamp	9B	8	Luigy	5B	1500000
18	Head Lamp	9B	3	Luigy	5B	1300000

3. **OR**
'Or' adalah operator yang digunakan untuk menampilkan data jika salah satu kondisi yang dipisahkan *'Or'* bernilai benar (TRUE).

KODE:

```sql
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2;
```

CONTOH

```sql
SELECT * FROM produk WHERE Tipe = 'Running' OR Tipe = 'Luigy';
```



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	Running	7B	50000
2	STOP Lamp	9B	5	Running	7B	500000
13	STOP Lamp	9B	8	Luigy	5B	1500000
18	Head Lamp	9B	3	Luigy	5B	1300000

4. **ORDER BY**

'Order By' adalah sebuah operator yang digunakan untuk mengurutkan hasil *query* berdasarkan nilai dalam satu atau beberapa kolom. Ada dua pengurutan, yang pertama adalah *Ascending (ASC)* dan yang kedua adalah *Descending (DSC)*. *ASC* digunakan untuk mengurutkan data dalam urutan terkecil ke terbesar, sedangkan *DSC* digunakan untuk mengurutkan data dalam urutan terbesar ke terkecil.

KODE:

sql   Salin kode

```sql
SELECT column1, column2, ...
FROM table_name
ORDER BY column1 [ASC|DESC];
```

CONTOH

ASCENDING

```sql
SELECT * FROM produk ORDER BY Stok ASC;
```



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
19	Blok Mesin	10B	1	BRT	3B	800000
20	Knalpot...	11B	1	Luster	6B	2000000
6	Knalpot...	15B	2	Samlong	7B	1000000
17	Speedo ...	12B	2	KOSO	4B	20000000
9	Head Lamp	9B	3	XO R...	10B	2000000
18	Head Lamp	9B	3	Luigy	5B	1300000
2	STOP Lamp	9B	5	Running	7B	500000
7	Velg Marus	10B	5	Bintang	7B	30000000
8	ECU	3B	5	RC M...	9B	8000000

DESCENDING

```sql
SELECT * FROM produk ORDER BY Stok DESC;
```



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
3	Speedo ...	12B	20	SIP	3B	1500000
4	Ban Pir...	11B	12	Diablo	6B	400000
1	Lampu Sein	10B	10	Running	7B	50000
12	Lowring...	11B	10	Marus	5B	500000
5	Ban Maxis	9B	8	Victra	6B	300000
13	STOP Lamp	9B	8	Luigy	5B	1500000
14	Kaliper	7B	8	RCB	3B	800000
10	Shockbr...	12B	6	Ohlins	6B	5000000
2	STOP Lamp	9B	5	Running	7B	500000

BAB 3
Operator

Operator adalah sebuah simbol yang digunakan untuk melakukan operasi perbandingan, aritmatika, dan pola pada database. Operator memungkinkan untuk melakukan berbagai tugas, seperti membandingkan nilai, melakukan perhitungan matematika, menggabungkan kondisi, dan lain-lain.

A. OPERATOR PERBANDINGAN

1. = (*Equal To*)

Operator ini digunakan untuk membandingkan apakah dua nilai sama.

KODE:

```sql
SELECT * FROM tabel WHERE kolom = nilai;
```

CONTOH

```sql
SELECT * FROM produk WHERE Harga = 500000;
```



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
2	STOP Lamp	9B	5	Running	7B	500000
12	Lowring...	11B	10	Marus	5B	500000

2. != atau <> (*Not Equal to)*

Operator ini digunakan untuk membandingkan apakah dua nilai tidak sama.

KODE:

```sql
SELECT * FROM tabel WHERE kolom != nilai;
```

CONTOH

```sql
SELECT * FROM produk WHERE Harga != 500000;
```



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	Running	7B	50000
3	Speedo ...	12B	20	SIP	3B	1500000
4	Ban Pir...	11B	12	Diablo	6B	400000
5	Ban Maxis	9B	8	Victra	6B	300000
6	Knalpot...	15B	2	Samlong	7B	1000000
7	Velg Marus	10B	5	Bintang	7B	30000000
8	ECU	3B	5	RC M...	9B	8000000
9	Head Lamp	9B	3	XO R...	10B	2000000
10	Shockbr...	12B	6	Ohlins	6B	5000000

3. < (*Less Than*)

Operator ini digunakan untuk membandingkan apakah nilai di sebelah kiri kurang dari nilai.

KODE:

```sql
SELECT * FROM tabel WHERE kolom < nilai;
```

CONTOH

```sql
SELECT * FROM produk WHERE Harga < 500000;
```



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	Running	7B	50000
4	Ban Pir...	11B	12	Diablo	6B	400000
5	Ban Maxis	9B	8	Victra	6B	300000

4. **> (*Greater Than*)**

Operator ini digunakan untuk membandingkan apakah nilai di sebelah kiri lebih besar dari nilai.

KODE:

```sql
SELECT * FROM tabel WHERE kolom > nilai;
```

CONTOH

```sql
SELECT * FROM produk WHERE Harga > 500000;
```



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
3	Speedo ...	12B	20	SIP	3B	1500000
6	Knalpot...	15B	2	Samlong	7B	1000000
7	Velg Marus	10B	5	Bintang	7B	30000000
8	ECU	3B	5	RC M...	9B	8000000
9	Head Lamp	9B	3	XO R...	10B	2000000
10	Shockbr...	12B	6	Ohlins	6B	5000000
11	Shockbr...	12B	5	KTC ...	12B	1700000
13	STOP Lamp	9B	8	Luigy	5B	1500000
14	Kaliper	7B	8	RCB	3B	800000

5. **<= (*Less Than or Equal to*)**

Operator ini digunakan untuk membandingkan apakah nilai di sebelah kiri kurang dari atau sama dengan nilai.

KODE:

```sql
SELECT * FROM tabel WHERE kolom <= nilai;
```

CONTOH

```sql
SELECT * FROM produk WHERE Harga <= 500000;
```



Kode_Produk	Nama_Sparepart	Stok	Tipe	Harga
1	Lampu Sein 10B	10	Running 7B	50000
2	STOP Lamp 9B	5	Running 7B	500000
4	Ban Pir... 11B	12	Diablo 6B	400000
5	Ban Maxis 9B	8	Victra 6B	300000
12	Lowring... 11B	10	Marus 5B	500000

6. **>= (*Greater Than or Equal to*)**

   Operator ini digunakan untuk membandingkan apakah nilai di sebelah kiri lebih besar dari atau sama dengan nilai.

   KODE:

```sql
SELECT * FROM tabel WHERE kolom >= nilai;
```

CONTOH

```sql
SELECT * FROM produk WHERE Harga >= 500000;
```



Kode_Produk	Nama_Sparepart	Stok	Tipe	Harga
2	STOP Lamp 9B	5	Running 7B	500000
3	Speedo ... 12B	20	SIP 3B	1500000
6	Knalpot... 15B	2	Samlong 7B	1000000
7	Velg Marus 10B	5	Bintang 7B	30000000
8	ECU 3B	5	RC M... 9B	8000000
9	Head Lamp 9B	3	XO R... 10B	2000000
10	Shockbr... 12B	6	Ohlins 6B	5000000
11	Shockbr... 12B	5	KTC ... 12B	1700000
12	Lowring... 11B	10	Marus 5B	500000

7. **BETWEEN**

Operator ini digunakan untuk memeriksa apakah nilai berada dalam suatu range tertentu.

```sql
SELECT * FROM tabel WHERE kolom BETWEEN nilai1 AND nilai2;
```

CONTOH

```sql
SELECT * FROM produk WHERE Harga BETWEEN 500000 AND 1000000;
```



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
2	STOP Lamp	9B	5	Running	7B	500000
6	Knalpot...	15B	2	Samlong	7B	1000000
12	Lowring...	11B	10	Marus	5B	500000
14	Kaliper	7B	8	RCB	3B	800000
16	MASTER Rem	10B	5	RCB	3B	1000000
19	Blok Mesin	10B	1	BRT	3B	800000

8. **LIKE**

Operator ini digunakan untuk mencocokan nilai dengan pola.

KODE:

```sql
SELECT * FROM tabel WHERE kolom LIKE 'nilai%';
```

CONTOH

```sql
SELECT * FROM produk WHERE Harga LIKE '5%';
```



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	Running	7B	50000
2	STOP Lamp	9B	5	Running	7B	500000
10	Shockbr...	12B	6	Ohlins	6B	5000000
12	Lowring...	11B	10	Marus	5B	500000

9. **IN**

Operator ini digunakan untuk mencocokan nilai dengan pola.

KODE:

```sql
SELECT * FROM tabel WHERE kolom IN (nilai1, nilai2, nilai3);
```

CONTOH

```sql
SELECT * FROM produk WHERE Nama_Sparepart IN ('Lampu Sein', 'Velg Marus', 'Knalpot Shijiro');
```



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	Running	7B	50000
6	Knalpot...	15B	2	Samlong	7B	1000000
7	Velg Marus	10B	5	Bintang	7B	30000000

10. **IS NULL / IS NOT NULL**

Operator ini digunakan untuk memeriksa apakah nilai adalah NULL atau tidak NULL

KODE:

```sql
SELECT * FROM tabel WHERE kolom IS NULL;
```

```sql
SELECT *
FROM your_table
WHERE your_column IS NOT NULL;
```

CONTOH

IS NULL

```
SELECT * FROM produk WHERE tipe IS NULL
```



Kode_Produk	Nama_Sparepart	Stok	Tipe	Harga

IS NOT NULL

```
SELECT * FROM produk WHERE tipe IS NOT NULL
```



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	Running	7B	50000
2	STOP Lamp	9B	5	Running	7B	500000
3	Speedo ...	12B	20	SIP	3B	1500000
4	Ban Pir...	11B	12	Diablo	6B	400000
5	Ban Maxis	9B	8	Victra	6B	300000
6	Knalpot...	15B	2	Samlong	7B	1000000
7	Velg Marus	10B	5	Bintang	7B	30000000
8	ECU	3B	5	RC M...	9B	8000000
9	Head Lamp	9B	3	XO R...	10B	2000000

B. OPERATOR ARITMATIKA

1. PENJUMLAHAN ATAU PENAMBAHAN (+)

sql Salin kode

```sql
SELECT kolom1 + kolom2 AS hasil_penambahan FROM nama_tabel;
```

sql Salin kode

```sql
UPDATE your_table
SET your_column = your_column + value_to_add
WHERE your_condition;
```

CONTOH

SELECT

```sql
SELECT Stok + Harga AS  hasil_penambahan FROM produk;
```



hasil_penambahan
50010
500005
1500020
400012
300008
1000002
30000005
8000005
2000003

UPDATE

```sql
UPDATE produk
SET harga = Stok + Harga
WHERE nama_sparepart = 'Lampu Sein';
```

Hasil:

Kode_Produk	Nama_Sparepart	Stok	Tipe	Harga
1	Lampu Sein 10B	10	Running 7B	50000



Kode_Produk	Nama_Sparepart	Stok	Tipe	Harga
1	Lampu Sein 10B	10	Running 7B	50010

2. PENGURANGAN (-)

```sql
SELECT kolom1 - kolom2 AS hasil_pengurangan FROM nama_tabel;
```

```sql
UPDATE your_table
SET your_column = your_column - value_to_subtract
WHERE your_condition;
```

CONTOH

SELECT

```sql
SELECT Harga - Stok AS  hasil_pengurangan FROM produk;
```



hasil_pengurangan
49990
499995
1499980
399988
299992
999998
29999995
7999995
1999997

UPDATE

```sql
UPDATE produk
SET harga = Harga - Stok
WHERE nama_sparepart = 'Lampu Sein';
```

Hasil:

Kode_Produk	Nama_Sparepart	Stok	Tipe	Harga
1	Lampu Sein 10B	10	Running 7B	50010



Kode_Produk	Nama_Sparepart	Stok	Tipe	Harga
1	Lampu Sein 10B	10	Running 7B	50000

3. PERKALIAN (*)

```sql
SELECT kolom1 * kolom2 AS hasil_perkalian FROM nama_tabel;
```

```sql
UPDATE your_table
SET your_column = your_column * multiplication_factor
WHERE your_condition;
```

CONTOH

SELECT

```sql
SELECT Harga * Stok AS  hasil_perkalian FROM produk;
```



hasil_perkalian
500000
2500000
30000000
4800000
2400000
2000000
150000000
40000000
6000000

UPDATE

```sql
UPDATE produk
SET harga = harga * stok
WHERE nama_sparepart = 'lampu sein'
```

Hasil:

Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	Running	7B	50000



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	Running	7B	500000

4. PEMBAGIAN (/)

```sql
SELECT kolom1 / kolom2 AS hasil_pembagian FROM nama_tabel;
```

```sql
UPDATE your_table
SET your_column = your_column / divisor
WHERE your_condition;
```

CONTOH

SELECT

```sql
SELECT Harga / Stok AS  hasil_pembagian FROM produk;
```



hasil_pembagian
5000.0000
100000.0000
75000.0000
33333.3333
37500.0000
500000.0000
6000000.0000
1600000.0000
666666.6667

UPDATE

```sql
UPDATE produk
SET harga = harga / stok
WHERE nama_sparepart = 'lampu sein'
```

Hasil:

Kode_Produk	Nama_Sparepart	Stok	Tipe	Harga
1	Lampu Sein 10B	10	Running 7B	500000



Kode_Produk	Nama_Sparepart	Stok	Tipe	Harga
1	Lampu Sein 10B	10	Running 7B	50000

C. OPERATOR POLA

1. LIKE 'a%'

Operator ini digunakan untuk mencari nilai apapun pada field nama yang dimulai dengan huruf 'a'.

KODE:

```sql
SELECT * FROM my_table WHERE column_name LIKE 'a%';
```

CONTOH

```sql
SELECT * FROM produk WHERE Nama_Sparepart LIKE 'b%';
```



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
4	Ban Pir...	11B	12	Diablo	6B	400000
5	Ban Maxis	9B	8	Victra	6B	300000
19	Blok Mesin	10B	1	BRT	3B	800000

2. LIKE '%a'

Operator ini digunakan untuk mencari nilai apapun pada field nama yang diakhiri dengan huruf 'a'

KODE:

```sql
SELECT * FROM my_table WHERE column_name LIKE '%a';
```

CONTOH

```sql
SELECT * FROM produk WHERE Nama_Sparepart LIKE '%p';
```



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
2	STOP Lamp	9B	5	Running	7B	500000
9	Head Lamp	9B	3	XO Raceboy	10B	2000000
13	STOP Lamp	9B	8	Luigy	5B	1500000
18	Head Lamp	9B	3	Luigy	5B	1300000

3. **LIKE '%or%'**
Operator ini digunakan untuk mencari nilai apapun pada field nama yang di dalamnya terdapat nama 'or'.

KODE:

```sql
SELECT * FROM my_table WHERE column_name LIKE '%or%';
```

CONTOH

```
SELECT * FROM produk WHERE Nama_Sparepart LIKE '%lamp%';
```



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	Running	7B	50000
2	STOP Lamp	9B	5	Running	7B	500000
9	Head Lamp	9B	3	XO Raceboy	10B	2000000
13	STOP Lamp	9B	8	Luigy	5B	1500000
18	Head Lamp	9B	3	Luigy	5B	1300000

4. **LIKE '_r%'**
Operator ini digunakan untuk mencari nilai apapun pada field nama yang karakter keduanya huruf 'r'.
KODE:

```sql
SELECT * FROM my_table WHERE column_name LIKE '_r%';
```

CONTOH

```
SELECT * FROM produk WHERE Nama_Sparepart LIKE '_a%';
```



Kode_Produk	Nama_Sparepart	Stok	Tipe	Harga
1	Lampu Sein 10B	10	Running 7B	50000
4	Ban Pir... 11B	12	Diablo 6B	400000
5	Ban Maxis 9B	8	Victra 6B	300000
14	Kaliper 7B	8	RCB 3B	800000
15	Kaliper 7B	5	Brembo 6B	6000000
16	MASTER Rem 10B	5	RCB 3B	1000000

5. LIKE ‘a_%’

Operator ini digunakan untuk mencari nilai apapun pada field nama yang dimulai dengan huruf ‘a’ dan panjangnya minimal 2 karakter.

KODE:

```sql
SELECT * FROM my_table WHERE column_name LIKE 'a_%';
```

CONTOH

```sql
SELECT * FROM produk WHERE Nama_Sparepart LIKE 'm_%';
```



Kode_Produk	Nama_Sparepart	Stok	Tipe	Harga
16	MASTER Rem 10B	5	RCB 3B	1000000

6. LIKE ‘a__%’

Operator ini digunakan untuk mencari nilai apapun pada field nama yang dimulai dengan huruf ‘a’ dan panjangnya minimal 3 karakter.

KODE:

```sql
SELECT * FROM my_table WHERE column_name LIKE 'a__%';
```

CONTOH

```sql
SELECT * FROM produk WHERE Nama_Sparepart LIKE 'm__%';
```



Kode_Produk	Nama_Sparepart	Stok	Tipe	Harga
16	MASTER Rem 10B	5	RCB 3B	1000000

7. **LIKE "a%o'**
Operator ini digunakan untuk mencari nilai apapun pada field yang dimulai dengan huruf 'a' dan diakhiri dengan huruf 'o'.

KODE:

```sql
SELECT * FROM my_table WHERE column_name LIKE 'a%o';
```

CONTOH

```
SELECT * FROM produk WHERE Nama_Sparepart LIKE 'L%N';
```



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	Running	7B	50000

BAB 4
Agregator

Agregator dalam *SQL* adalah sebuah fungsi *SQL* yang digunakan untuk melakukan perhitungan pada kumpulan data. Agregator digunakan dalam perintah ***Select*** maupun ***Update***.

Contoh Perintah Agregator:

1. **SUM**
Menghitung jumlah nilai dalam satu kolom.

```sql
SELECT SUM(nama_kolom) FROM nama_tabel;
```

CONTOH

```
SELECT SUM(Harga) AS harga_total FROM produk;
```



harga_total
84350000

2. AVG

Untuk menghitung nilai rata-rata nilai dalam suatu kolom.

sql Salin kode

```
SELECT AVG(nama_kolom) FROM nama_tabel;
```

CONTOH

```
SELECT AVG(Harga) AS harga_rata_rata FROM produk;
```

⬇

harga_rata_rata
4217500.0000

3. COUNT

Untuk menghitung jumlah baris atau nilai dalam suatu kolom.

sql Salin kode

```
SELECT COUNT(nama_kolom) FROM nama_tabel WHERE kondisi;
```

CONTOH

```
SELECT COUNT(*) AS total FROM produk;
```

⬇

total
20

4. MAX

Untuk menemukan nilai maksimum dalam suatu kolom.

```sql
SELECT MAX(nama_kolom) FROM nama_tabel;
```

CONTOH
`SELECT MAX(Harga) AS harga_tertinggi FROM produk;`



harga_tertinggi
30000000

5. MIN

Untuk menemukan nilai minimum dalam suatu kolom.

```sql
SELECT MIN(nama_kolom) FROM nama_tabel;
```

CONTOH
`SELECT MIN(Harga) AS harga_terendah FROM produk;`



harga_terendah
50000

6. GROUP_CONCAT

Untuk menggabungkan nilai-nilai dalam suatu kolom berdasarkan grup, dan mengembalikan hasil dalam bentuk string yang digabungkan.

```sql
SELECT GROUP_CONCAT(nama_kolom SEPARATOR ', ')
FROM nama_tabel
GROUP BY kolom_grup;
```

CONTOH

```sql
SELECT GROUP_CONCAT(CONCAT(Harga, ', ', Stok) SEPARATOR ', ') AS 'Harga dan Stok'
FROM produk
GROUP BY Harga;
```



Harga dan Stok	
50000, 10	9B
300000, 8	9B
400000, 12	10B
500000, 10, 500...	21B
800000, 8, 8000...	20B
1000000, 2, 100...	22B
1300000, 3	10B
1500000, 20, 15...	23B
1700000, 5	10B
2000000, 3, 200...	22B
5000000, 6	10B
6000000, 5	10B
8000000, 5	10B
[illegible]	[illegible]

7. STDDEV dan VARIANCE

Untuk menghitung deviasi standar dan varians dari suatu kolom, masing-masing.

```sql
SELECT STDDEV(nama_kolom) FROM nama_tabel;
SELECT VARIANCE(nama_kolom) FROM nama_tabel;
```

CONTOH

STDDEV `SELECT STDDEV(harga) FROM produk;`  stddev(harga) 7383652.1282
VARIANCE `SELECT VARIANCE(stok) FROM produk;`  variance(stok) 19.0600

BAB 5

Data Definition Language (*DDL*)

Data Definition Language (*DDL*) adalah bagian dari *Structured Query Language* (*SQL*) yang digunakan untuk mendefinisikan struktur *database*. Pernyataan *DDL* digunakan untuk membuat, mengubah, dan menghapus objek *database* termasuk tabel, tampilan, indeks, dan prosedur tersimpan.

Berikut adalah perintah *DDL*:

1. **Create Database**
   'Create Database' adalah sebuah perintah dalam *SQL* yang digunakan untuk membuat sebuah *database* baru di sistem manajemen basis data.

KODE:

sql Salin kode

```sql
CREATE DATABASE nama_basis_data;
```

CONTOH
`CREATE DATABASE catatan_penjualan_toko_sparepart_motor`

2. **Create Table**
   'Create Table' adalah sebuah perintah dalam *SQL* yang digunakan untuk membuat tabel baru di dalam *database*. Tabel adalah struktur penyimpanan data yang terdiri dari

baris dan kolom, di mana setiap kolom memiliki tipe data tertentu dan setiap baris merepresentasikan sebuah record data.

KODE:

```sql
CREATE TABLE nama_tabel (
    kolom1 Tipe_Data1 [Batasan1],
    kolom2 Tipe_Data2 [Batasan2],
    ...
    [CONSTRAINT constraint_name FOREIGN KEY (kolom_fk) REFERENCES nama_tabel_referensi (kolom
);
```

CONTOH

```sql
CREATE TABLE produk(
Kode_Produk INT,
Nama_Sparepart TEXT,
Stok INT,
Tipe TEXT,
Harga INT);
```

```sql
CREATE TABLE transaksi(
kode_produk INT,
kode_transaksi INT,
jumlah_produk INT,
tanggal DATE);
```

3. **Drop Table**
'Drop Table' adalah sebuah perintah dalam *DDL* yang digunakan untuk menghapus tabel dari database. Ketika sebuah tabel dihilangkan / di '*drop*' maka semua data yang ada di dalam tabel tersebut akan hilang secara permanen.

KODE:

```sql
DROP TABLE nama_tabel;
```

CONTOH

```sql
DROP TABLES transaksi;
```

Hasil:

Tables_in_catatan_penjualan_toko_sparepart_motor
produk
transaksi



Tables_in_catatan_penjualan_toko_sparepart_motor
produk

4. **Alter Table**

'Alter Table' adalah sebuah perintah dalam *DDL* yang digunakan untuk merenovasi (menambah kolom baru, menghapus kolom baru, dan mengubah batasan pada kolom) atau *rename* sebuah entitas.

KODE:

```sql
ALTER TABLE nama_tabel RENAME TO nama_baru;
```

Kode diatas digunakan untuk mengubah nama tabel.

```sql
ALTER TABLE nama_tabel
ADD nama_kolom Tipe_Data;
```

Kode diatas digunakan untuk menambahkan kolom baru.

```sql
ALTER TABLE nama_tabel
DROP COLUMN nama_kolom;
```

Kode diatas digunakan untuk menghapus kolom.

CONTOH

```sql
ALTER TABLE PRODUK RENAME TO BARANG;
```

Hasil:

Tables_in_catatan_penjualan_toko_sparepart_motor
produk

⬇

Tables_in_catatan_penjualan_toko_sparepart_motor
barang

```sql
ALTER TABLE produk
ADD COLUMN merk_barang VARCHAR(255);
```

Hasil:

Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	Running	7B	50000
2	STOP Lamp	9B	5	Running	7B	500000
3	Speedo ...	12B	20	SIP	3B	1500000
4	Ban Pir...	11B	12	Diablo	6B	400000
5	Ban Maxis	9B	8	Victra	6B	300000
6	Knalpot...	15B	2	Samlong	7B	1000000
7	Velg Marus	10B	5	Bintang	7B	30000000



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga	merk_barang
1	Lampu Sein	10B	10	Running	7B	50000	(NULL)
2	STOP Lamp	9B	5	Running	7B	500000	(NULL)
3	Speedo ...	12B	20	SIP	3B	1500000	(NULL)
4	Ban Pir...	11B	12	Diablo	6B	400000	(NULL)
5	Ban Maxis	9B	8	Victra	6B	300000	(NULL)

```
ALTER TABLE produk
DROP COLUMN merk_barang;
```

Hasil:

Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga	merk_barang
1	Lampu Sein	10B	10	Running	7B	50000	(NULL)
2	STOP Lamp	9B	5	Running	7B	500000	(NULL)
3	Speedo ...	12B	20	SIP	3B	1500000	(NULL)
4	Ban Pir...	11B	12	Diablo	6B	400000	(NULL)
5	Ban Maxis	9B	8	Victra	6B	300000	(NULL)



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	Running	7B	50000
2	STOP Lamp	9B	5	Running	7B	500000
3	Speedo ...	12B	20	SIP	3B	1500000
4	Ban Pir...	11B	12	Diablo	6B	400000
5	Ban Maxis	9B	8	Victra	6B	300000
6	Knalpot...	15B	2	Samlong	7B	1000000
7	Velg Marus	10B	5	Bintang	7B	30000000

BAB 6
Data Manipulation Language (DML)

Data Manipulation Language (*DML*) adalah sekumpulan perintah SQL yang digunakan untuk memanipulasi data dalam skema yang dibuat oleh DDL. Ini berkaitan dengan data aktual dan digunakan untuk memasukkan, memperbarui, dan mengambil data dari database. Contoh pernyataan DML termasuk SELECT, INSERT, UPDATE, dan DELETE.

Berikut adalah perintah *DML*:

1. **Select**

   'Select' adalah perintah dalam *SQL* yang digunakan mengambil data dari sebuah tabel dalam database. *Select* dapat digunakan secara sendiri untuk mengambil beberapa kolom dari sebuah tabel, atau dapat digabungkan dengan klausa lainnya seperti ***'from'*** dan ***'where'***.

KODE:

```sql
SELECT column1, column2, ...
FROM table_name;
```

CONTOH
SELECT * FROM PRODUK



Kode_Produk	Nama_Sparepart	Stok	Tipe	Harga
1	Lampu Sein 10B	10	Running 7B	50000
2	STOP Lamp 9B	5	Running 7B	500000
3	Speedo ... 12B	20	SIP 3B	1500000
4	Ban Pir... 11B	12	Diablo 6B	400000
5	Ban Maxis 9B	8	Victra 6B	300000
6	Knalpot... 15B	2	Samlong 7B	1000000
7	Velg Marus 10B	5	Bintang 7B	30000000
8	ECU 3B	5	RC M... 9B	8000000
9	Head Lamp 9B	3	XO R... 10B	2000000

Jika kita melakukan ***select*** menggunakan simbol bintang '*' maka semua kolom yang terdapat dalam tabel tersebut akan muncul.

```
SELECT Nama_Sparepart, Harga FROM PRODUK
```



Nama_Sparepart	Harga
Lampu Sein 10B	50000
STOP Lamp 9B	500000
Speedo ... 12B	1500000
Ban Pir... 11B	400000
Ban Maxis 9B	300000
Knalpot... 15B	1000000
Velg Marus 10B	30000000
ECU 3B	8000000
Head Lamp 9B	2000000

Tetapi jika kita melakukan ***select*** dengan hanya mencantumkan beberapa nama kolom yang terdapat dalam tabel, maka kolom yang akan muncul hanya kolom yang kita select saja.

2. Insert Into

'Insert Into' dalam *SQL* digunakan untuk menambahkan atau menyisipkan data baru kedalam sebuah tabel. Data ini dimasukan kedalam tabel dengan nilai yang sudah ditentukan oleh pengguna.

KODE:

```sql
INSERT INTO nama_tabel (kolom1, kolom2, kolom3, ...)
VALUES (nilai1, nilai2, nilai3, ...);
```

CONTOH

```sql
INSERT INTO produk (Kode_Produk, Nama_Sparepart, Stok, Tipe, Harga)
VALUES
(1, 'Lampu Sein', 10, 'Running', 50000),
(2, 'STOP Lamp', 5, 'Running', 500000),
(3, 'Speedo Meter', 20, 'SIP', 1500000),
(4, 'Ban Pirelli', 12, 'Diablo', 400000),
(5, 'Ban Maxis', 8, 'Victra', 300000),
(6, 'Knalpot Shijiro', 2, 'Samlong', 1000000),
(7, 'Velg Marus', 5, 'Bintang', 30000000),
```

Hasil:

Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	Running	7B	50000
2	STOP Lamp	9B	5	Running	7B	500000
3	Speedo ...	12B	20	SIP	3B	1500000
4	Ban Pir...	11B	12	Diablo	6B	400000
5	Ban Maxis	9B	8	Victra	6B	300000
6	Knalpot...	15B	2	Samlong	7B	1000000
7	Velg Marus	10B	5	Bintang	7B	30000000

3. **Update**
   'Update' adalah perintah yang digunakan untuk memperbarui data yang sudah ada dalam sebuah tabel dalam database. Perintah ini dapat mengubah nilai-nilai kolom yang ada dalam tabel.

KODE:

```sql
UPDATE table_name
SET column1 = value1, column2 = value2, ...;
```

Perintah diatas digunakan untuk memperbaharui data secara masal pada setiap atribut di entitas tanpa memperdulikan kondisi

```sql
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Perintah diatas digunakan untuk memperbaharui data pada baris tertentu dalam sebuah entitas dengan menggunakan kondisi.

CONTOH

Update tanpa menggunakan klausa ***Where***

```
UPDATE produk SET tipe = 'running'
```

Hasil:

Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	Running	7B	50000
2	STOP Lamp	9B	5	Running	7B	500000
3	Speedo ...	12B	20	SIP	3B	1500000
4	Ban Pir...	11B	12	Diablo	6B	400000
5	Ban Maxis	9B	8	Victra	6B	300000
6	Knalpot...	15B	2	Samlong	7B	1000000
7	Velg Marus	10B	5	Bintang	7B	30000000



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	running	7B	50000
2	STOP Lamp	9B	5	running	7B	500000
3	Speedo ...	12B	20	running	7B	1500000
4	Ban Pir...	11B	12	running	7B	400000
5	Ban Maxis	9B	8	running	7B	300000
6	Knalpot...	15B	2	running	7B	1000000
7	Velg Marus	10B	5	running	7B	30000000
8	ECU	3B	5	running	7B	8000000
9	Head Lamp	9B	3	running	7B	2000000

Update dengan menggunakan klausa ***Where***

```
UPDATE produk SET tipe = 'Diablo'
WHERE nama_sparepart = 'Lampu Sein';
```

Hasil:

Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	running	7B	50000
2	STOP Lamp	9B	5	running	7B	500000
3	Speedo ...	12B	20	running	7B	1500000
4	Ban Pir...	11B	12	running	7B	400000
5	Ban Maxis	9B	8	running	7B	300000
6	Knalpot...	15B	2	running	7B	1000000
7	Velg Marus	10B	5	running	7B	30000000
8	ECU	3B	5	running	7B	8000000
9	Head Lamp	9B	3	running	7B	2000000



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	Diablo	6B	50000
2	STOP Lamp	9B	5	running	7B	500000
3	Speedo ...	12B	20	running	7B	1500000
4	Ban Pir...	11B	12	running	7B	400000
5	Ban Maxis	9B	8	running	7B	300000
6	Knalpot...	15B	2	running	7B	1000000
7	Velg Marus	10B	5	running	7B	30000000
8	ECU	3B	5	running	7B	8000000
9	Head Lamp	9B	3	running	7B	2000000

4. **Delete**
   'Delete' adalah sebuah perintah dalam *DML* yang digunakan untuk menghapus baris atau data dalam sebuah database.

KODE:

```sql
DELETE FROM table_name;
```

Perintah diatas digunakan untuk menghapus semua semua baris dari tabel yang ditentukan.

```sql
DELETE FROM table_name
WHERE condition;
```

Perintah diatas digunakan untuk menghapus baris-baris dari tabel yang memenuhi kondisi yang ditentukan.

CONTOH

Delete tanpa menggunakan klausa ***Where***

```
DELETE FROM produk;
```

Hasil:

Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	Diablo	6B	50000
2	STOP Lamp	9B	5	running	7B	500000
3	Speedo ...	12B	20	running	7B	1500000
4	Ban Pir...	11B	12	running	7B	400000
5	Ban Maxis	9B	8	running	7B	300000
6	Knalpot...	15B	2	running	7B	1000000
7	Velg Marus	10B	5	running	7B	30000000
8	ECU	3B	5	running	7B	8000000
9	Head Lamp	9B	3	running	7B	2000000



Kode_Produk	Nama_Sparepart	Stok	Tipe	Harga

Delete dengan menggunakan klausa ***Where***

```
DELETE FROM produk WHERE nama_sparepart = 'lampu sein';
```

Hasil:

Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
1	Lampu Sein	10B	10	Diablo	6B	50000
2	STOP Lamp	9B	5	running	7B	500000
3	Speedo ...	12B	20	running	7B	1500000
4	Ban Pir...	11B	12	running	7B	400000
5	Ban Maxis	9B	8	running	7B	300000
6	Knalpot...	15B	2	running	7B	1000000
7	Velg Marus	10B	5	running	7B	30000000
8	ECU	3B	5	running	7B	8000000
9	Head Lamp	9B	3	running	7B	2000000



Kode_Produk	Nama_Sparepart		Stok	Tipe		Harga
2	STOP Lamp	9B	5	running	7B	500000
3	Speedo ...	12B	20	running	7B	1500000
4	Ban Pir...	11B	12	running	7B	400000
5	Ban Maxis	9B	8	running	7B	300000
6	Knalpot...	15B	2	running	7B	1000000
7	Velg Marus	10B	5	running	7B	30000000
8	ECU	3B	5	running	7B	8000000
9	Head Lamp	9B	3	running	7B	2000000
10	Shockbr...	12B	6	running	7B	5000000

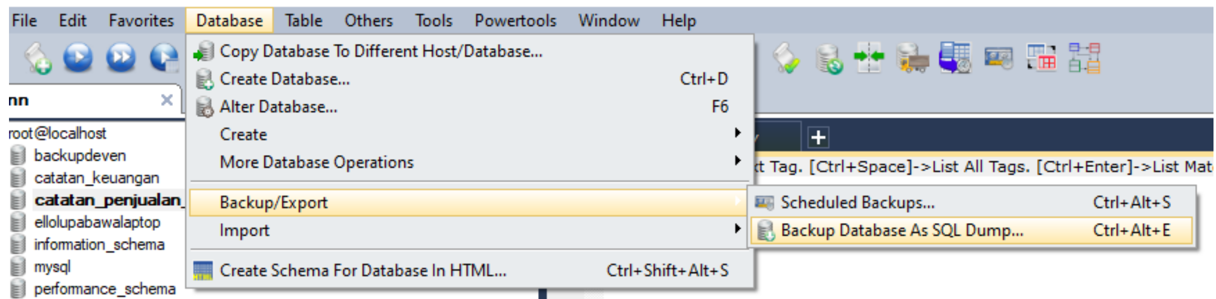
BAB 7

CARA MELAKUKAN BACKUP & RESTORE DATABASE PADA APLIKASI SQLYOG

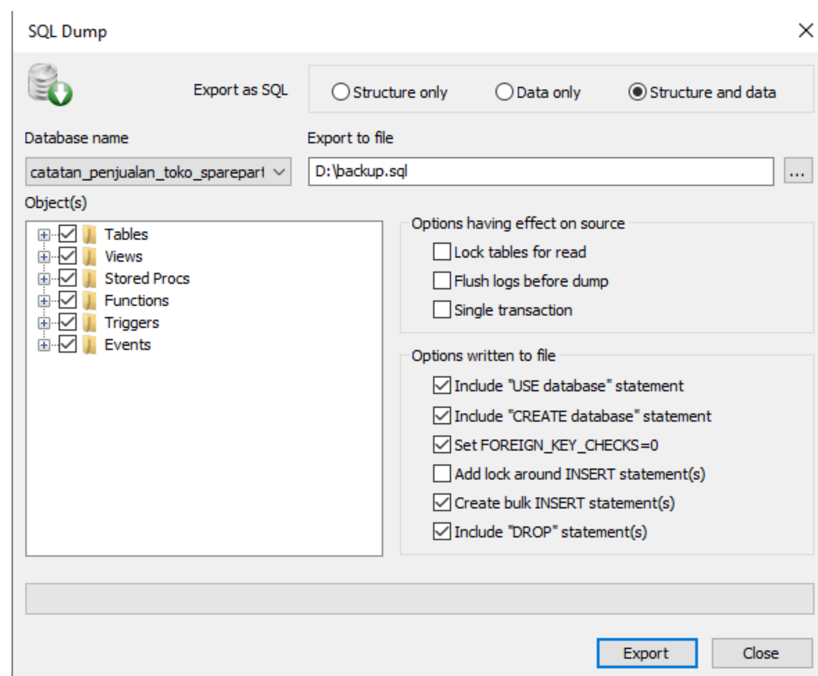
Backup memiliki tujuan untuk menyelamatkan data. Sedangkan restore memiliki tujuan untuk mengembalikan data maupun sistem sama seperti keadaan awalnya. Jika *database* anda rusak karena beberapa alasan, maka *file backup* bisa menjadi cadangan untuk mengembalikan data yang hilang.

1. **Cara Untuk Melakukan Backup:**

- **Langkah pertama** dalam melakukan *backup* adalah, klik bagian '*database*' pada taskbar. Kemudian klik pilihan '*backup/export*', lalu pilih '*Backup Database As SQL Dump…*'

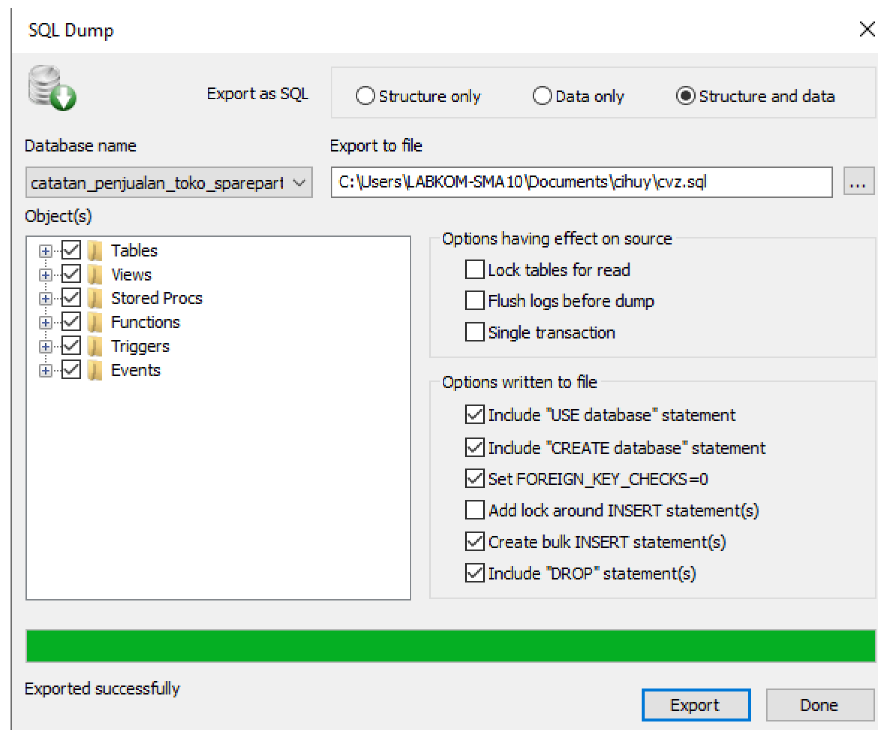


- **Langkah Kedua**
  Pilih *database* mana yang ingin di *backup* dan tentukan dimana file akan di *save*, kemudian berikan nama untuk file tersebut. Setelah selesai klik tombol '***EXPORT***'.



-

- **Langkah ketiga,** setelah muncul tulisan '*Exported succesfully*' klik tombol '***DONE***'



2. Cara Untuk Melakukan Restore:

- **Pertama,** Buatlah sebuah *database* baru
- **Kedua**, *use* database yang baru saja dibuat
- **Ketiga**, buka file *database* yang sudah anda *backup* lalu copy semua kodenya dan di *paste* ke *SQLYOG*. Kemudian blok semua kode tersebut lalu klik F8 pada *keyboard*. Tunggu beberapa saat hingga code berhasil dijalankan.
- *Restore* sudah berhasil dilakukan dan *database* sudah bisa digunakan.

DAFTAR PUSTAKA

Amalia, Rifka. 2023
https://www.gamelab.id/news/2687-structured-query-language-sql-pengertian-jenis-jenis-dan-fungsinya

Indobot Academy. 2024
https://indobot.co.id/pengertian-dml-ddl-dan-dcl-pada-mysql/

Kuncoro, Ari Arsito. 2022
https://teknik-informatika-s1.stekom.ac.id/informasi/baca/Pengertian-DDL-dan-DML-Beserta-Contoh-Perintahnya-dalam-Database/764d4cc8477ab0f54654ba0e80a71e687a3f987c

Dqlab. 2023
https://dqlab.id/mengenal-command-ddl-dml-dcl-dan-tcl-dalam-mysql

Taufiq. 2020
https://www.infanthree.com/pengertian-ddl-dml-dan-sql/